DTU Library

The Next Generation Platform as A Service: Composition and Deployment of Platforms and Services

Kentis, Angelos Mimidis; Soler, José; Veitch, Paul; Broadbent, Adam; Mobilio, Marco; Riganelli, Oliviero; Van Rossem, Steven; Tavernier, Wouter; Sayadi, Bessem

Next Generation Platform as a Service: Composition and Deployment of Platforms and Services

Angelos Mimidis[1]*, Jose Soler[1], Paul Veitch [2], Adam Broadbent [2], Marco Mobilio[3], Oliviero Riganelli[3], Steven Van Rossem[4], Wouter Tavernier[4], Bessem Sayadi[5]

[1] Technical University of Denmark; agmimi@fotonik.dtu.dk, joss@fotonik.dtu.dk

[2] British Telecom; paul.veitch@bt.com, adam.broadbent@bt.com

[3] University of Milano Bicocca; marco.mobilio@disco.unimib.it, oliviero.riganelli@unimib.it

[4] University of Ghent; steven.vanrossem@ugent.be, wouter.tavernier@ugent.be

[5] Nokia Bell Labs France; bessem.sayadi@nokia-bell-labs.com

* Correspondence: agmimi@fotonik.dtu.dk

Abstract: The emergence of widespread cloudification and virtualisation promises increased flexibility, scalability, and programmability for the deployment of services by Vertical Service Providers (VSPs). This cloudification also improves service and network management, reducing the Capital and Operational Expenses (CAPEX, OPEX). A truly cloud-native approach is essential, since 5G will provide a diverse range of services - many requiring stringent performance guarantees while maximising flexibility and agility despite the technological diversity. This paper proposes a workflow based on the principles of build-to-order, Build-Ship-Run, and automation; following the Next Generation Platform as a Service (NGPaaS) vision. Through the concept of Reusable Functional Blocks (RFBs), an enhancement to Virtual Network Functions, this methodology allows a VSP to deploy and manage platforms and services, agnostic to the underlying technologies, protocols, and APIs. To validate the proposed workflow, a use case is also presented herein, which illustrates both the deployment of the underlying platform by the Telco operator and of the services that run on top of it. In this use case, the NGPaaS operator facilitates a VSP to provide Virtual Network Function as a Service (VNFaaS) capabilities for its end customers.

Keywords: 5G, Cloud, Cloud-native, PaaS, NFV, SDN, Telco-grade, CORD

1. Introduction

5G is expected to be the ubiquitous fabric to provide reliable, low-latency and high-speed connectivity for a wide variety of services (IoT, mobile, Industry 4.0, automotive, etc.). However current network architectures are not yet able to provide these characteristics, mainly due to their monolithic nature which limits their flexibility, scalability, and programmability. To facilitate the adoption of 5G, the consensus in the industry is to utilise technologies and workflows from the field of Information Technology (IT), such as the cloudification and virtualisation of services [1]. This way, network services that previously comprised dedicated and monolithic hardware appliances (e.g.

38   Firewalls) will now be virtualised instances in commodity servers in the cloud. Doing so is expected
39   to offer a much higher degree of flexibility, scalability, and automation to service providers, as they
40   will be able to modify the virtual setups dynamically and programmatically.

41   To fully reap the benefits provided by virtualisation and cloudification, the deployment and
42   management of services must be decoupled from the underlying computing and networking
43   infrastructure. A paradigm that can facilitate this requirement is the Platform as a Service (PaaS), as
44   it provides a centralised and reliable environment, with high degrees of agility and automation, on
45   which services can be deployed and managed. The benefits provided by the PaaS model can be
46   further enhanced, by incorporating the paradigms of Software Defined Networking (SDN) and
47   Network Function Virtualization (NFV). The scope of SDN would be to facilitate the control and
48   programmability of the network infrastructure while of NFV to allow for the efficient virtualisation
49   and management of services on commodity servers.

50   In contrast with IT services, 5G services are more diverse and stringent in terms of their network
51   and computing requirements. These requirements imply the need for a more flexible and
52   customizable PaaS model [2], than the currently available non 5G oriented solutions. One such model
53   is the Next Generation Platform as a Service (NGPaaS) [3] [4], which is built around the principles of
54   micro-services, modularity, and build-to-order. In short, NGPaaS allows for the deployment of
55   custom-built platforms and services, on a diverse set of infrastructure technologies, using automated
56   and technology agnostic workflows. Through the same workflow, any Vertical Service Provider
57   (VSP) (e.g. Telco provider, IoT provider) can deploy either a custom platform on the available
58   infrastructure or services on top of an existing platform.

59   The content of this paper is structured as follows. At first, a study of related work is presented
60   in Section 2. Section 3 presents the implementation of the requirements and architecture of NGPaaS,
61   as conceptualised in [3], [4] - specifically, it provides the NGPaaS architectural design, the concept of
62   Reusable Functional Blocks (RFBs) and introduces the RFB Description and Composition Languages
63   Design, Deploy and Direct (RDCL 3D) tool [5]. Later in Section 4, the NGPaaS workflow is practically
64   validated through a use case. This use case (Telco PaaS) is based on the Central Office Re-architected
65   as a Datacenter (CORD) [6] platform and provides with Virtual Network Function (VNF) as a Service
66   (VNFaaS) capabilities to a VSP. In addition to describing the use case and the components that
67   comprise it, this paper also details the process of how these components were made NGPaaS-ready
68   in Section 5. Later, in Section 6 a brief description of another use case targeted by NGPaaS (5G Public
69   Safety) is provided Finally, Section 7 provides a critical conclusion of the presented work.

2. Related Work

71   As mentioned in the introduction, most telco-grade platform environments have originated
72   from existing IT-based cloud technologies (e.g. OpenStack, Kubernetes), but have been extended with
73   new features to be suitable for telco environments, such as edge, access, and core networks. The scope
74   of this section is to contextualise telco-grade platform solutions with recent PaaS and NFV related
75   work.

76   The most prominent standardisation effort in the NFV platform area is the ETSI NFV
77   Management and Orchestration (MANO) specification [7], which details an NFV-based platform
78   architecture. There, a fixed set of PaaS related features is coupled tightly to the underlying
79   infrastructure managers. However, unlike NGPaaS, ETSI NFV MANO lacks the option to customise
80   which specific PaaS components are included in a deployment, consequently, the supported
81   virtualisation technologies and infrastructure types cannot be modified easily or swapped. Instead,
82   the MANO functionality focuses on the lifecycle management of specific services, controlled by pre-
83   integrated infrastructure managers and PaaS functions. Different flavours of MANO platforms are
84   available (e.g. OSM [19] or 5GTANGO [20]), and each one may have proprietary interfaces and

85 service descriptor formats, which can act as a barrier to the integration of other PaaS solutions. The
86 authors in [23] and [24] show that the characteristics might differ significantly when comparing
87 existing MANO solutions. It may be required, depending on the targeted use-case, to augment an
88 existing MANO platform and integrate new functionalities ([23], [25]).

89   In addition to MANO platforms, there are many advances are being made in the NFV
90 technology space. One such example is in addition to common x86 server hardware, the use of FPGAs
91 in data centre infrastructure is being presented as virtualised resources, as demonstrated in [26]. In
92 [27], a management framework is presented for using FPGA resources in data centres, also enabling
93 FPGAs to run Virtual Network Functions (VNFs). The authors of [28] have implemented a framework
94 to design network functions running on split CPU and FPGA architectures -this provides a method
95 to offload computationally-intensive processing of the network functions to the FPGA.

96   The development of improved algorithms to assist in the deployment and operation of
97 virtualised network functions is a highly active research area. A method for customised VNF
98 placement is proposed in 5G networks [29], showing that due to the diverse performance
99 requirements among different 5G scenarios, an adaptive VNF placement approach is needed to
100 accommodate to service-specific requirements automatically. In addition to placement, scaling
101 algorithms can also enhance the operation of VNFs. In [30], a formal method is described to predict
102 VNF traffic and optimally scale the VNF resources accordingly in an elastic way. Similarly, in [33], a
103 novel analytical model based on Stochastic Network Calculus is presented to investigate the end-to-
104 end performance bound of chained VNFs quantitatively.   By modelling (non-)bursty network traffic,
105 the proposed analytical model provides an efficient method for service providers to determine which
106 service chaining and placement configuration are more beneficial to meet the SLA requirements in
107 terms of violation error and latency performance. Another operational enhancement is an optimised
108 load balancing system for VNFs, presented in [31] and the placement of multiple VNFs in geo-
109 distributed infrastructures is investigated in [32]. The latter is particularly applicable in a telecom
110 scenario, where an optimal choice must be made in which central offices the VNFs should run.

111   In additional to VNF placement and scaling, automated healing actions also need to be
112 implemented to quickly analyse and solve any issues with the running network service before
113 performance is degraded. Autonomous healing actions in [34] are based on pre-defined workflows
114 developed by experts. The primary idea behind the adopted approach is to transfer the current expert
115 domain knowledge and manual action triggers into a rule-based self-healing solution. Moreover, in
116 [35], it is shown that deep learning can be used to identify anomaly events from NFV system logs
117 reliably. A technique using a supervised machine learning method for online classification of
118 anomaly states based on similarities between anomaly type-specific density grid patterns is presented
119 in [36]. The detection is done by analysing CPU, memory and network usage; - the procedure used,
120 the density grid mapping, is inherited from the theory of grid-based clustering. Operational
121 procedures such as healing, combine techniques from various research areas and gather information
122 from a wide range of data, implying the integration of a diverse set of tools and libraries. The above-
123 described references show that the PaaS should cater for quick and easy upgradeability, in order to
124 incorporate the rapid evolutions happening in the NFV area and to allow easy customisation for a
125 Telco or other use-case. The platform should follow a modular approach, and a relevant PaaS
126 architecture for this method is advised in [3]. Later, in section 4, we exemplify this modular setup by
127 showing how we implemented in NGPaaS the functional blocks of the orchestration mechanism,
128 network control, monitoring framework and the analysis of VNF anomalies to trigger healing actions.

129   According to [8], a unified PaaS interface can be derived from existing PaaS Application
130 Programming Interfaces (APIs), or put another way, a common set of functionalities regarding
131 service instantiation and configuration can be grouped under a standard API. This allows for a
132 unified interface for application deployment and management, among different cloud platforms,
133 avoiding technology lock-in effects. Unfortunately, considerable effort is needed to maintain and

134 translate the unified API to the proprietary PaaS APIs; a common trade-off between the overhead
135 and benefits of using an API abstraction layer. In contrast to [8], NGPaaS adopts a workflow-based
136 orchestration mechanism [3], hence the API or descriptor format of the targeted PaaS or infrastructure
137 manager is addressed natively in the workflow, thus minimising the effort required to integrate
138 support for new APIs.

139 The NGPaaS architecture is innovative in the sense that both PaaS and service components are
140 considered as the same module type, using the same generic orchestration mechanism based on the
141 RFB model. As argued in [9], if software suppliers adopt the generic RFB descriptor model, DevOps-
142 based integration strategies are greatly facilitated. The NGPaaS framework, therefore, facilitates
143 customisation and the creation of a custom PaaS solution, with functionality targeted at specific use
144 cases**.** Given the vast range of possible services, it can be challenging to integrate all existing options
145 into a one-size-fits-all platform solution. NGPaaS implements the complete service lifecycle using
146 build-ship-run actions: 1) What is needed in terms of service offering (build), 2) How the service can
147 be deployed (ship) and 3) Where the service should be deployed (run). These fundamental design
148 questions facilitate the mapping of the right PaaS technologies to the available IaaS and the
149 requirements of the service that should be executed. Each PaaS can be enhanced with unique features
150 related to network state, Quality of Service, high availability, auto-scaling or SDK toolsets [9]. The
151 deployed set of PaaS features is then customizable and use-case dependent. Before exemplifying this
152 through the Telco PaaS use case, we first elaborate on the different processes and workflows which
153 enable this modular NGPaaS framework.

3. The Next Generation PaaS: Architecture, Concepts, Processes, and Workflows

155 The scope of this section is to provide the reader with an understanding of the NGPaaS. More
156 specifically this section introduces the overall NGPaaS architecture, the concept of RFBs, how RFB
157 graphs can be composed via the RDCL 3D tool and finally introduces the workflow through which
158 platforms and services can be composed and deployed in NGPaaS.

3.1 The NGPaaS architecture

160 NGPaaS consist of a multi-layer architecture, with each layer being responsible for a specific
161 functionality. In total, six layers are defined, 1) The Business Registration Layer, 2) The Business as a
162 Service (BaaS) Layer 3) The Business and Operation Support System (BSS/OSS) Layer, 4) The Dev-
163 for-operations Layer, 5) The Platform as a Service (PaaS) Layer and finally 6) The Infrastructure as a
164 Service (IaaS) Layer. The details and scope of each of these layers can be found in **Table 1**, while
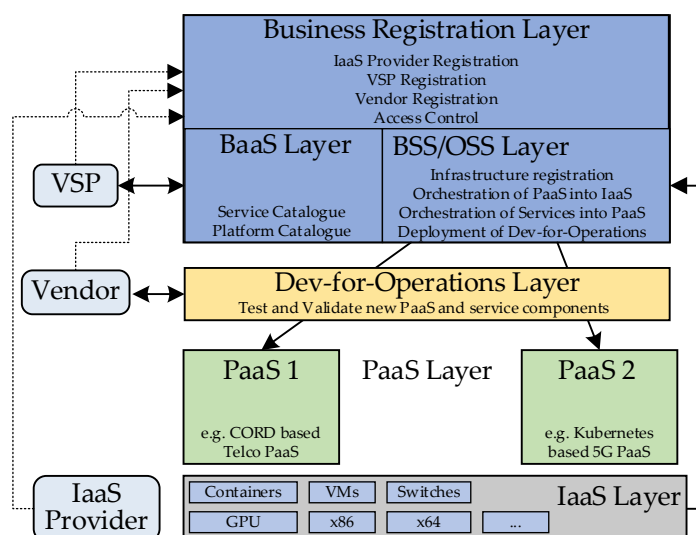165 **Figure 1** provides a simplified view of the NGPaaS architecture and also of the cross-layer
166 interactions.

Figure 1: Simplified view of the NGPaaS architecture

Table 1: Layers of the NGPaaS architecture

NGPaaS Layer	Description
Business Registration	Registers all stakeholders that participate in NGPaaS (VSPs, Vendors, etc). It also includes the resolution of access and execution rights.
BaaS	A customizable catalogue from which to order service or platform workloads.
BSS/OSS	Responsible for inventory registration, global supervision, and deployment of services and platforms on their execution environments. In our prototype, the BSS/OSS role [12] is fulfilled by the RDCL 3D tool.
Dev-for-Operations	An environment to support the innovative NGPaaS Dev-for-Operations model - an evolution of the existing DevOps model to facilitate new types of interaction and development methods between multiple stakeholders. This environment is where staging and development are performed to bring new PaaS or service components into NGPaaS [13].
PaaS	PaaS components are deployed on the available (declared) infrastructure. These PaaS components form the framework to manage the VNFs included in the requested services. More than one PaaS instances can be active and managed at the same time by the OSS/BSS.
IaaS	This layer relates to the cloud infrastructure available to the NGPaaS operator.

3.2. Reusable Functional Blocks

RFBs are a core concept of NGPaaS, as they have a significant influence in its architecture and workflows. RFBs are a logical representation of functions that can decompose a complex system into simple sub-functions. In a sense, RFBs are a generalisation of the Virtual Network Function (VNF) concept proposed by ETSI [10]. There are however some core differences between ETSI VNFs and NGPaaS RFBs. The main difference is that RFBs are recursive and thus can be arbitrarily decomposed into other RFBs, while VNFs in the ETSI model cannot be decomposed into other VNFs (the composition stops at the level of VNF Component). In addition, RFBs can be mapped into both

178 software and hardware execution environments [10], while VNFs can only be mapped into Virtual
179 Machines (or Containers) in traditional cloud infrastructures. RFBs can also be paired with metadata
180 during their composition, which allows for high degrees of configuration, with reduced complexity.
181 **Figure 2** illustrates the RFB concept. Additionally, NGPaaS is a higher-level orchestration framework
182 when compared to ETSI NFV, as it facilitates the deployment and management of multiple
183 heterogeneous platforms. For example, it could be the case that an Open Source MANO (OSM)
184 platform, a reference implementation of ETSI NFV, is deployed and managed by NGPaaS. A detailed
185 comparison between the ETSI NFV and NGPaaS proposal is also provided at [3].
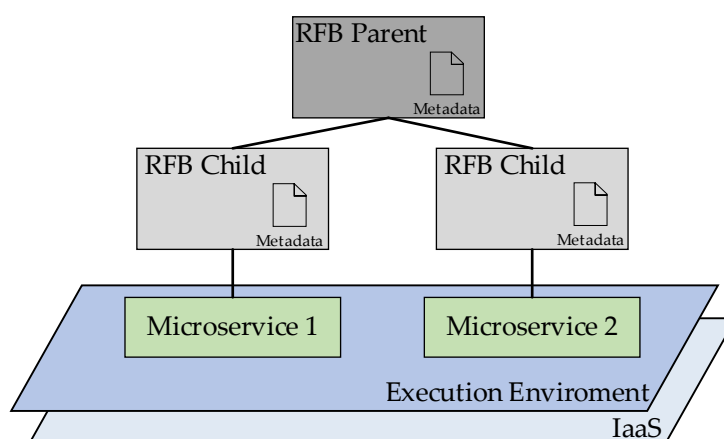
186

187　　　　　　　　　　　　　　　Figure 2: The concept of Reusable Functional Blocks

188 The figure shows how an RFB parent is decomposed into two RFB children, each of which is mapped
189 to the deployment of a micro-service into the execution environment. For these RFBs to be properly
190 configured, they are each paired with some metadata. The NGPaaS RFB model supports inheritance
191 from an RFB parent to its children; this means that, unless overwritten by an RFB child, metadata
192 from the parent will be propagated to all its children.

193 *3.2. RFB Description and Composition Languages Design, Deploy and Direct tool*

194 To fully exploit the RFB concept, NGPaaS is utilising the RDCL 3D tool: a web-based framework
195 that allows the composition of RFB graphs. By composing graphs comprising multiple RFBs, it is
196 possible for a VSP to define complex platforms and services for deployment. Apart from composing
197 RFB graphs, the RDCL 3D tool is also responsible for "shipping" them to an RDCL agent. Every
198 RDCL agent is tied to an execution environment (e.g. an infrastructure or a platform) and its role is
199 to deploy the services/platforms composed by the graph into this execution environment. To facilitate
200 this action, each RFB leaf in an RFB graph is mapped to an Ansible-based [22] workflow which is
201 executed onto the execution environment of the RFB by the agent. By selecting different execution
202 environments for different RFBs of an RFB graph, it is possible for a deployment to span multiple
203 execution environments. For example, a platform deployment can span across a public and private
204 cloud infrastructure**.** For the RFBs to be executed in the desired order, a priority mechanism is used
205 in which each RFB is assigned a priority value. The RDCL 3D tool was initially developed in the 5G-
206 PPP Superfluidity [11] project but has been heavily extended to meet the requirements of NGPaaS.
207 **Figure 3** illustrates the operation of the RDCL 3D tool in greater detail.
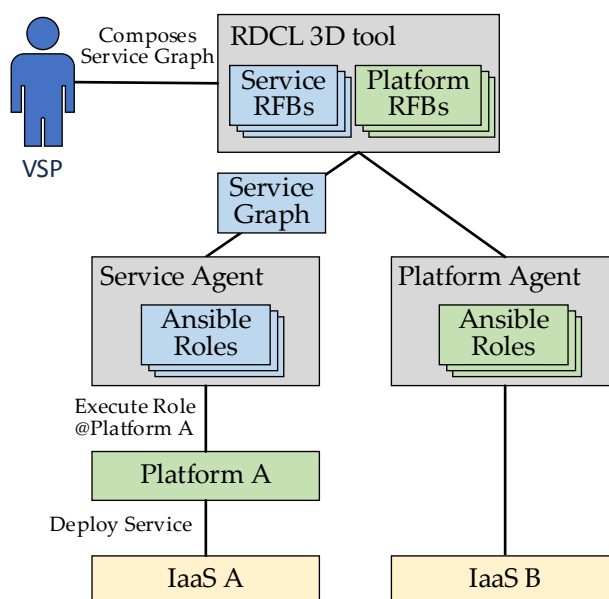
208

209

210

Figure 3: Overview of the RDCL 3D tool

A VSP uses the web-based interface of the RDCL 3D tool, to describe the desired service or platform using an RFB graph; this comprises of RFBs from the catalogue of RFBs made available by the NGPaaS operator. The RDCL 3D tool provides a full separation between platform and service deployments by assigning each RFB to either a service or platform category. Then when composing a service or platform graph, the VSP will be only presented with the corresponding RFBs. Once the composition of the RFB graph is completed, the VSP selects an appropriate RDCL agent (In this case the service agent of IaaS A). The RFB graph is encoded as a JSON string and is shipped to the agent. Once the agent receives the deployment request, it initiates the execution of the corresponding Ansible roles, each mapping to an individual RFB leaf of the RFB graph. Finally, these Ansible roles are executed into the appropriate execution environment (Platform A in this case) and the desired service is provisioned. Figure 3 also illustrates the possibility to control multiple RDCL agents, via a single RDCL 3D instance.

Figure 4 depicts a simple example of an RFB graph as it appears in the RDCL 3D tool web interface. A root RFB (service1_fb) comprises two RFB children (Service A, Service B) each corresponding to a different logical subcomponent of the parent service. Each of these RFB comprises of two RFB leaves (VNF A, VNF B, VNF C, and VNF D) -these leaf RFBs are the ones directly mapped to the deployment micro-services (e.g. containers) in the execution environment. Finally, the figure also shows as an example the metadata tied to leaf RFB VNF B (two key-value pairs).
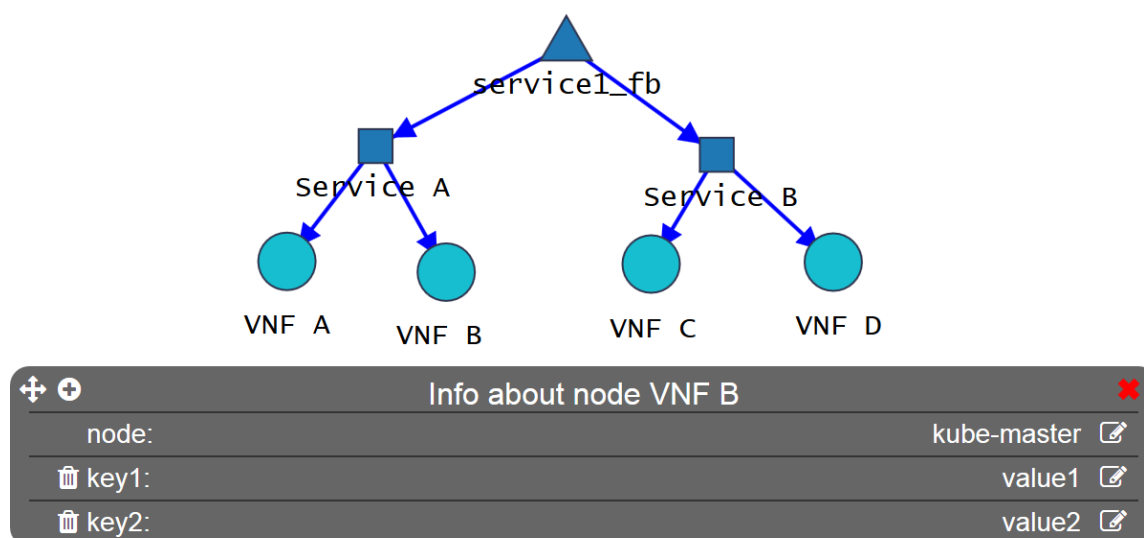
229

Figure 4: Example of RFB graph from the RDCL 3D tool

3.3. Processes and Workflows

The ability to deploy, in a modular way, both platform and service components is one of the key features of NGPaaS. This allows the creation of an eco-system where telco operators can easily cooperate and integrate with multiple software vendors. By using an aligned descriptor format between the vendor and operator side, such as the RFB model, it becomes easier to share both service components and their related execution environments (PaaS). The processes defined in this section, enables the deployment of a customizable set of PaaSes and service components, empowering the support of many use-cases. The most critical processes between layers mentioned above are illustrated in **Figure 5** and also analysed in **Table 2**.

Table 2: Processes of the NGPaaS architecture

NGPaaS Process	Details
Infrastructure registration	A generic provisioning process to support the broad spectrum of available infrastructure technologies. According to the use-case and the required services, the appropriate infrastructure nodes are leased and registered in the OSS.
PaaS orchestration	Refers to the deployment of selected PaaS components on the appropriate IaaS. The PaaS could be Kubernetes, CORD, or any other platform. The PaaS components can be aggregated flexibly and modelled as RFBs. This capability could not be implemented using ETSI MANO which is focused only on VNFs and assumes the platform is already deployed.
Service orchestration	A service is provided by deploying VNFs as sets of RFBs. The target execution environment of each VNF is included in the metadata of the related RFBs, which also specify the runtime aspect: VM, container, Unikernel, FPGA Bitstream, etc. The execution environment of a service component is a pre-deployed PaaS which supports the runtime aspects of the RFB.

Onboarding new components	The design rule in NGPaaS is "everything is an RFB" (VNF, ancillary services like orchestration, SDN controller, etc.). Therefore, all the components could be updated, upgraded, swapped, etc. This can be done through the usage of the Dev-for-Operations processes.

Before requesting any platform or service deployment, (1) the required Infrastructure for the VSP must be reserved by the NGPaaS operator. Once the appropriate infrastructure has been registered, then, (2) the VSP can request the deployment of the desired service. This is done by the composition of two RFB graphs, one for the underlying platform (if not already present) and one for the service itself. Once the RFB graphs have been composed then PaaS (3), or service workloads (4) can be orchestrated on their corresponding execution environments. A relatively similar workflow (5)-(8) is also available to software vendors through their dedicated Dev-for-Operations layer, which allows them (5)-(7) to test new software components that can be later (8) included as RFBs in the available RFB catalogue.
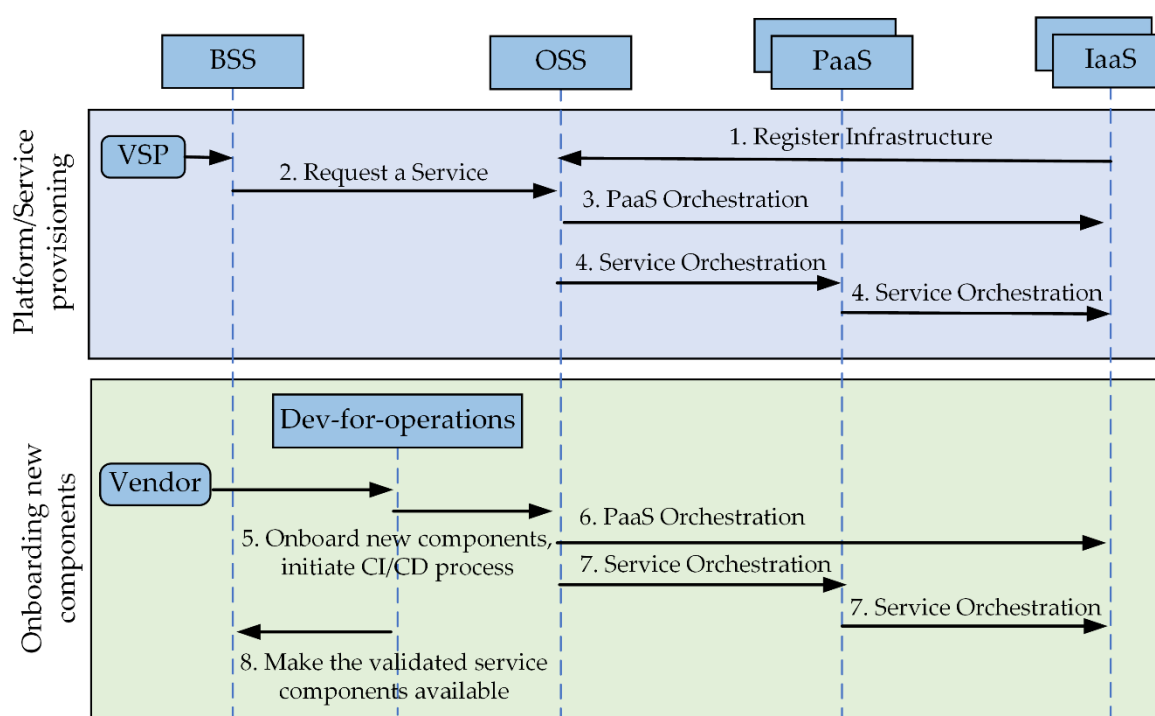

Figure 5: NGPaaS processes, to enable a customizable platform

4. Virtual Network Function as a Service, Overview

The scope of this section is to illustrate the operation of the NGPaaS workflow, via a demonstrable use case. It outlines a service scenario called "VNF-as-a-Service" (VNFaaS), which is based on the CORD platform. More specifically, this section highlights the composition of both the platform and services as RFB graphs and their subsequent deployment in their corresponding execution environment (IaaS for CORD, PaaS for the services).

4.1 The Telco PaaS Use Case: Virtual Network Function as a Service (VNFaaS)

In the Telco PaaS use case, the NGPaaS operator (e.g. a Tier 1 Telco operator) hosts VNFs on behalf of a Service Provider (the VSP). These VNFs could be virtual Routers and Firewalls, which the Service Provider can configure accordingly depending on the desired service (business VPNs,

263 Internet access, etc.). In a "pre-NGPaaS" NFV architecture, the on-boarding of VNFs, to be made
264 available in the service catalogue, is done by the Telco operator (based on the preferences of the
265 Service Provider).

266 Using the NGPaaS workflow,, the Telco operator instead utilises a PaaS-oriented approach
267 instead. While the Telco provider still deploys the platforms, the Service Providers can onboard and
268 administer specific VNFs on their own (e.g. vRouter, vFirewall) and also enable Value-added Service
269 (VAS) capabilities. These VASs could be Telco-grade enhancements to the basic service (e.g.
270 monitoring, healing, policy-based network control, etc.). This way, a pre-existing "VNF App Store"
271 can be supplemented by allowing the Service Provider to onboard new VNFs via direct interactions
272 with preferred vendors, resulting in a more diverse set of capabilities than could be obtainable with
273 the pre-NGPaaS model. There many benefits associated with the adoption of the VNFaaS use case
274 for both the VSP and its end customers. Most of these benefits are due to the nature of the VNFaaS
275 use case, for example, the Service Provider does not need to ship hardware to the premises of its end
276 customer allowing for a more flexible and scalable business and service model.

277 **Figure 6** illustrates the VNFaaS use case in more detail. There, a VSP has deployed two service graphs
278 through the RDCL 3D tool, each comprising of two VNFs. Deploying these two graphs has resulted
279 in the deployment of two service chains onto the CORD platform. Both service chains comprise an
280 interconnected virtual router and firewall and are associated with a unique end user. The figure also
281 illustrates the possibility to onboard and deploy VNFs of different vendors, thus providing flexibility
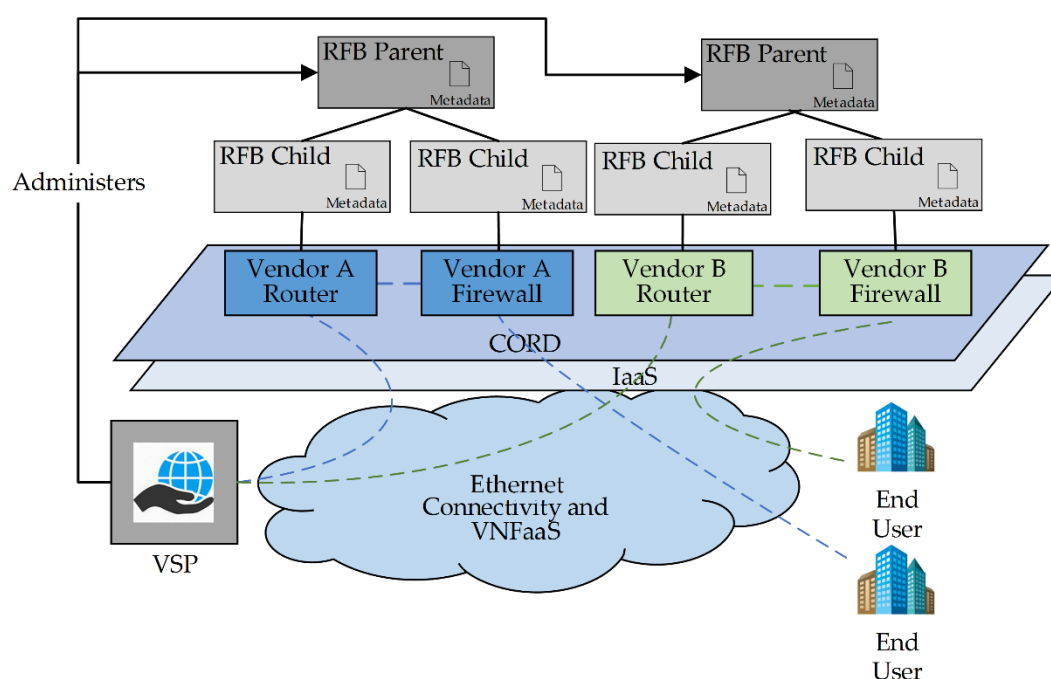282 to the end user.

283

284 Figure 6: The Virtual Network Function as a Service use-case

285 *4.2. The VNFaaS Proof of Concept*

286 To validate the NGPaaS workflow through the VNFaaS use case, a Proof of Concept (PoC)
287 scenario was designed and implemented. To be considered successful this PoC must be able to
288 showcase both the provisioning of a Telco-grade platform and the provisioning of services related to
289 the VNFaaS use case.

290   **Figure 7** illustrates an overview of the desired final status of the PoC. There, a Service Provider
291   has been granted access to the RDCL 3D tool by the NGPaaS operator. Using this tool, the Service
292   Provider is then able to compose and deploy service graphs of virtual routers and firewalls, which
293   will provide the desired service to the provider's end customers. Through the NGPaaS workflow,
294   these service graphs are then translated to platform-specific APIs and passed down to the platform
295   orchestrator. Since a service graph request will usually comprise both compute and network
296   resources, the platform orchestrator will translate the request to API calls targeting both the
297   underlying VIM and SDNC components of the platform. The VIM will then provision the required
298   virtual computing resources and the SDNC will provide the required connectivity. It should also be
299   possible for the service provider to monitor its deployment for specific KPIs. At the same time, an
300   alerting function should continuously check the collected data and trigger automated healing
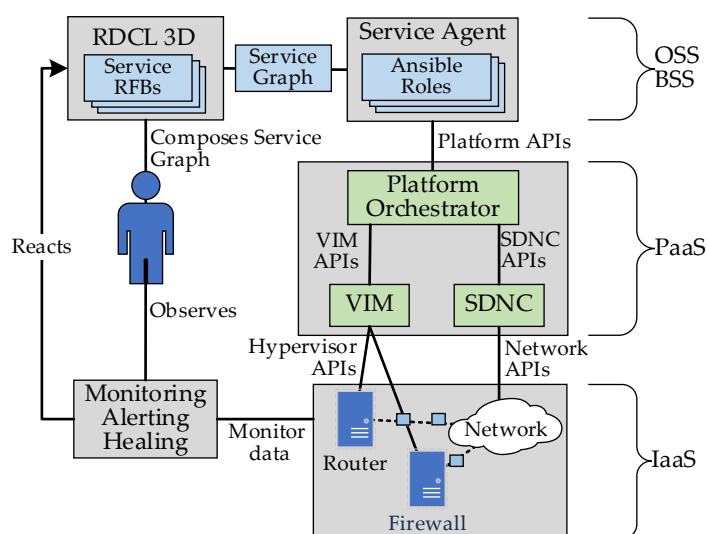301   workflows, when the state of a deployed service is undesirable.

302

303   Figure 7: VNFaaS Proof of Concept

304   The next couple of sections will provide more details on how this PoC was developed. More
305   specifically, the technologies that comprise it will be detailed (e.g. Platform of choice), together with
306   the steps required to integrate them into the NGPaaS workflow.

307   *4.3. The Telco PaaS Platform: Central Office Re-architected as a Datacenter (CORD)*

308   The selected platform for the Telco PaaS use case is CORD and the current prototype uses
309   version 6.0 [6]. The primary reason for selecting CORD was that it was designed to provide edge
310   network access, by acting as a virtualised Central Office. This implies that the core functionalities of
311   the CORD platform could be effectively reused to fit the needs of the VNFaaS use case. For example,
312   other platforms (e.g. OSM), have a more generic implementation which would make their integration
313   into the VNFaaS use case a more complex task.

314   CORD's architecture comprises three main functional elements, implemented as open source
315   projects: (1) The XOS orchestrator, responsible for the joined control of ONOS and OpenStack. This
316   joined control is facilitated by well-defined service models, which describe the supported services
317   and service synchronizers. The synchronizer's role is to align the operational state of CORD with the
318   desired state, as defined by the administrator, in a programmatic way. (2) The ONOS SDN Controller
319   (SDNC), responsible for managing the physical and virtual networks. (3) The OpenStack platform,
320   responsible for managing the lifecycle and resources of the deployed VNFs. In the used version of
321   CORD, all the individual CORD components are deployed as sets of Docker containers, managed by
322   Kubernetes and Helm. **Figure 8** illustrates a high-level overview of the CORD architecture.
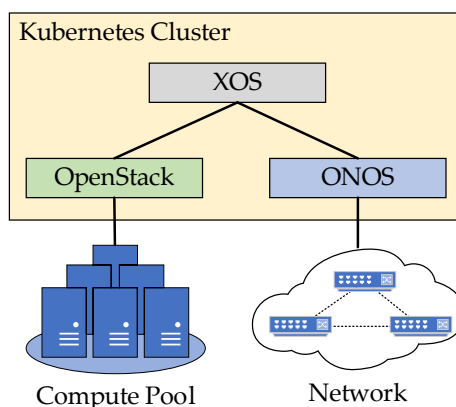
323

Figure 8: High-level overview of the CORD architecture

324

325   *4.4. Supported Virtual Network Functions (VNFs) and Value Added Services (VAS)*

326   4.4.1. Core Services: Virtual Firewalls and Virtual Routers

327   This section gives a brief overview of the VNFs and VASs supported by the Telco PaaS
328   prototype. As part of the experimental set-up, we use commercially available router and firewall
329   VNFs. The Router VNF is an enterprise-class router designed to run on x86 standard servers but
330   comprising all the same features as equivalent hardware routers [15]. Similarly, the Firewall VNF is
331   a virtual appliance aimed at enterprise-level security deployments [16]. The implementation of both
332   VNFs is based on virtual machines and in the presented prototype their running instances are hosted
333   by OpenStack.

334   4.4.2. Value Added Services: Monitoring, Alerting, and Healing

335   In the Telco PaaS, monitoring utilises the ElasticSearch-Logstash-Kibana (ELK) stack [14]. This
336   stack consists of three main components: (1) Elastic Search: A distributed search and analytics engine
337   based on REST APIs. (2) Logstash: A server-side data processing tool that can ingest data from
338   various probes and manipulate it before sending it to ElasticSearch. (3) Kibana: A visualisation tool
339   provided with ElasticSearch. As noted, in the Telco PaaS prototype the selected VNFs are based on
340   proprietary VMs, thus deploying monitoring probes internal to the VNFs is not feasible. To overcome
341   this limitation the probes in the Telco PaaS are deployed in a side-car fashion, externally to the target
342   service; but can poll the target service for the required using the available interfaces. For this
343   prototype, we rely on the SNMP protocol and REST interface to monitor the router VNF and ONOS
344   SDNC respectively. More specifically, SNMP is used to poll the virtual router for its CPU utilisation,
345   while REST is used to poll ONOS for network traffic information regarding the virtual interfaces of
346   the deployed VNFs.

347   As an enhancement to baseline monitoring, this prototype also provides Alerting and Healing
348   capabilities. The ELK stack can map trends in the CPU utilisation of monitored VNFs to predefined
349   anomalous behaviours. Once such an anomaly is detected, a Healing function is called, which
350   redeploys the anomalous VNF with more resources (e.g. more virtual CPUs or RAM). All healing
351   actions are performed, by the Healing rule, via the RDCL 3D tool, thus ensuring alignment between
352   the state of the deployment and the view of the tool. **Figure 9** shows the operation of the monitoring
353   VAS in detail. As mentioned before, monitoring, alerting, and healing are complementary services
354   provided selectively to VSPs. As a result, upon deployment of a VNF, the VSP can set a flag in the
355   RDCL 3D metadata to trigger or not the monitoring capabilities for this VNF.
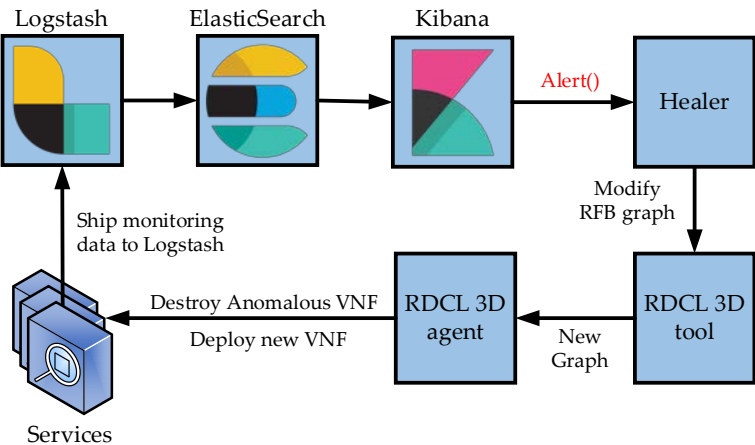
Figure 9: Operation of the Monitoring Value Added Service

Figure 10 illustrates the capability for live network traffic information for each interface of the provisioned VNFs. As mentioned, this information is collected by the monitoring probe through the ONOS REST API. On the other hand, **Figure 11** shows how live information is reported about the CPU utilisation of all monitored VNFs. In addition, this segment is configured with three predefined thresholds (50%, 80%, and 90%) - when any of these thresholds is reached a corresponding alert is shown to the administrator. In the provided example the critical threshold of 90% has been exceeded (94%); exceeding this threshold triggers an automated healing workflow that will re-provision the VNF with more virtual resources.
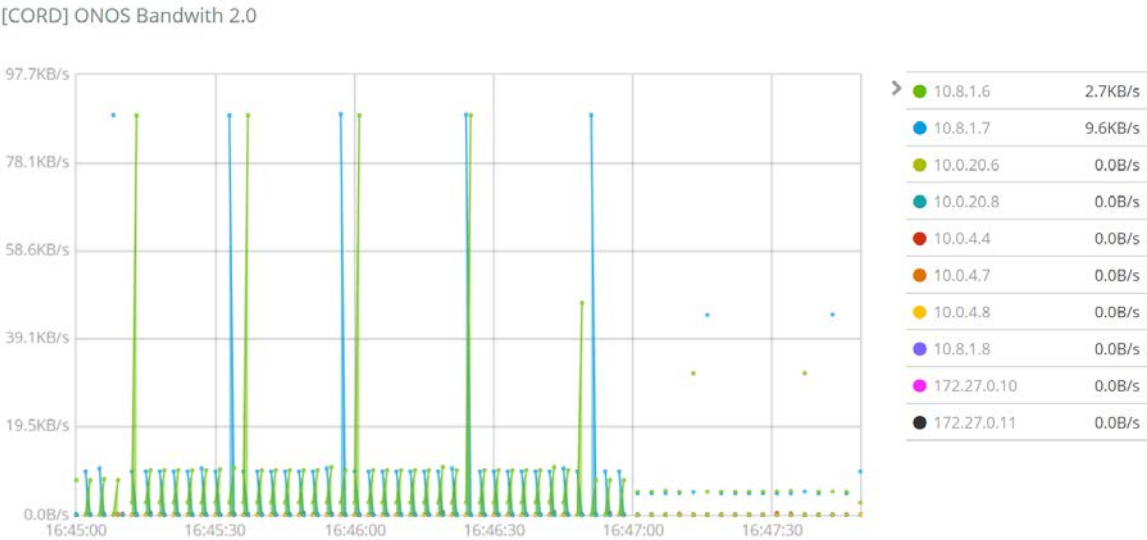

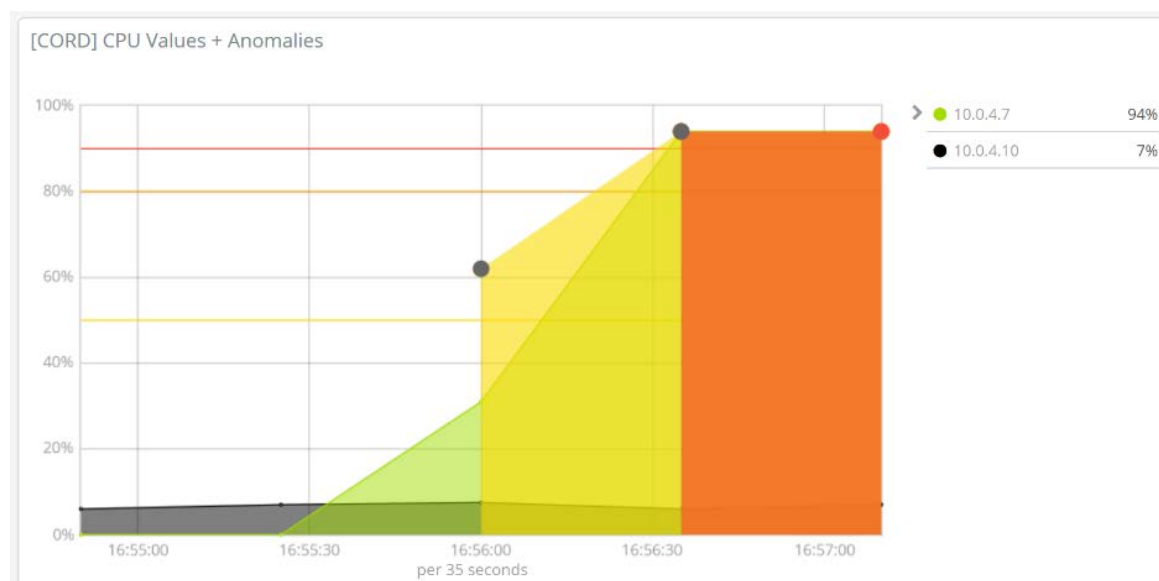Figure 10: Network Monitoring Dashboard of Telco PaaS

Figure 11: CPU Monitoring/Healing Dashboard of Telco PaaS

4.4.3. Value Added Services: Policy-Based Network Management

In addition to monitoring/alerting/healing, the Telco PaaS supports one more VAS, namely policy-based network management. This is achieved with the integration of an SDN-based network policy framework [18] to the ONOS SDNC. By acting as a layer of abstraction between the network infrastructure and the VSP, the network policy framework allows for simplified control over the network infrastructure. Also, the network policy framework follows an entirely modular architecture, consisting of a Policy Manager and multiple separate Policy Type applications, thus allowing the VSP to add support for new policy types on demand, without affecting the runtime operation of the policy framework. **Figure 12** illustrates a (simplified) example operation of the policy framework. There an administrator has deployed the policy manager with two policy type implementations (Firewall and NAT). Subsequently, the administrator issues a request to create a Firewall policy instance that will drop traffic between hosts H1 and H2. Upon receiving the request, the policy manager will validate it for correctness, and if successful, it will then request the Firewall Policy Type implementation to enforce the policy in the network. This will result in the Firewall Policy Type to install flow rules in the switches adjacent to H1 and H2, which will block traffic between the two endpoints.
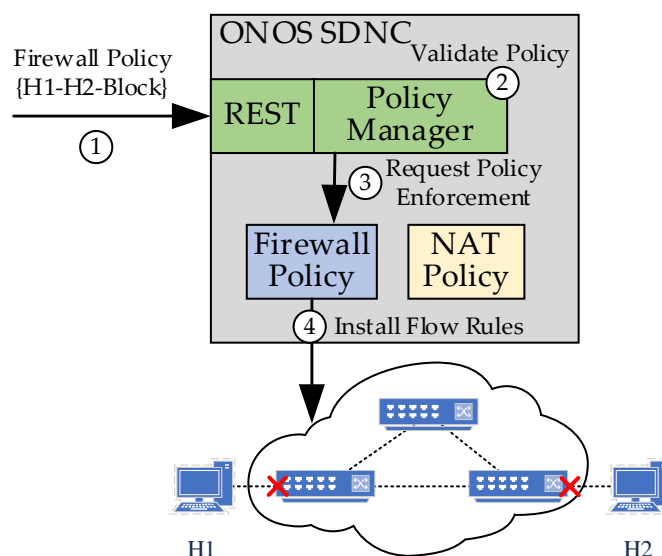
386

387   Figure 12: Operation of the Network Policy Framework

388   **5. Virtual Network Function as a Service: Component RFBization**

389   The scope of this section is to highlight how the different platform and service components that
390   comprise the Telco PaaS use case, have been mapped to individual RFBs and RFB graphs.

391   *5.1 RFBization of the CORD platform*

392   In previous versions of the CORD platform (<6.0), there was a very tight integration between the
393   different components of CORD, since the majority of them were executed either as native services or
394   as standalone containers and virtual machines. This translated to a very rigid deployment process,
395   with little to no flexibility. However, in the latest version (6.0), the deployment of CORD is now
396   managed via higher-level tools like Kubernetes [21] and Helm [17]. Through these tools, deploying,
397   managing and versioning the different CORD components is easier, more flexible and robust than
398   before. This new deployment approach greatly facilitated the process of RFBizing the CORD
399   platform, mainly due to the central point of management offered by the Kubernetes and Helm tools.

400   Deploying CORD via the RDCL 3D tool requires the definition of several RFBs, each of which
401   implements a specific functionality during the CORD deployment process. However, despite their
402   differences, all RFBs can be placed in one of three categories Pre-deployment configuration RFBs,
403   Deployment RFBs and Post-deployment configuration RFBs. **Table 3** provides details these
404   categories.

405   Table 3: RFB Categories

RFB Category	Details
Pre-deployment Configuration RFBs	Responsible for configuring the target infrastructure node for the subsequent deployment of the CORD platform. An example is the cloning of remote source code repositories so that CORD can be later built from.
Deployment RFBs	These are the core RFBs, that when executed will provide with a functional CORD deployment on the infrastructure node. The majority of them are tied to the deployment of groups of containers, through the Helm tool. An example is the deployment of the XOS orchestrator or the ONOS SDNC.

	These configure a CORD deployment for a specific use case or resolve possible issues that might have risen during the deployment of the CORD platform. An example is the registration of the CORD compute node to the ONOS SDNC.
Post-deployment configuration RFBs	

406

407 To reduce the complexity of the graph, only the Deployment RFBs are graphically illustrated by
408 the RDCL 3D tool, while the rest is passed to the agent as metadata. **Figure 13** illustrates the RFB
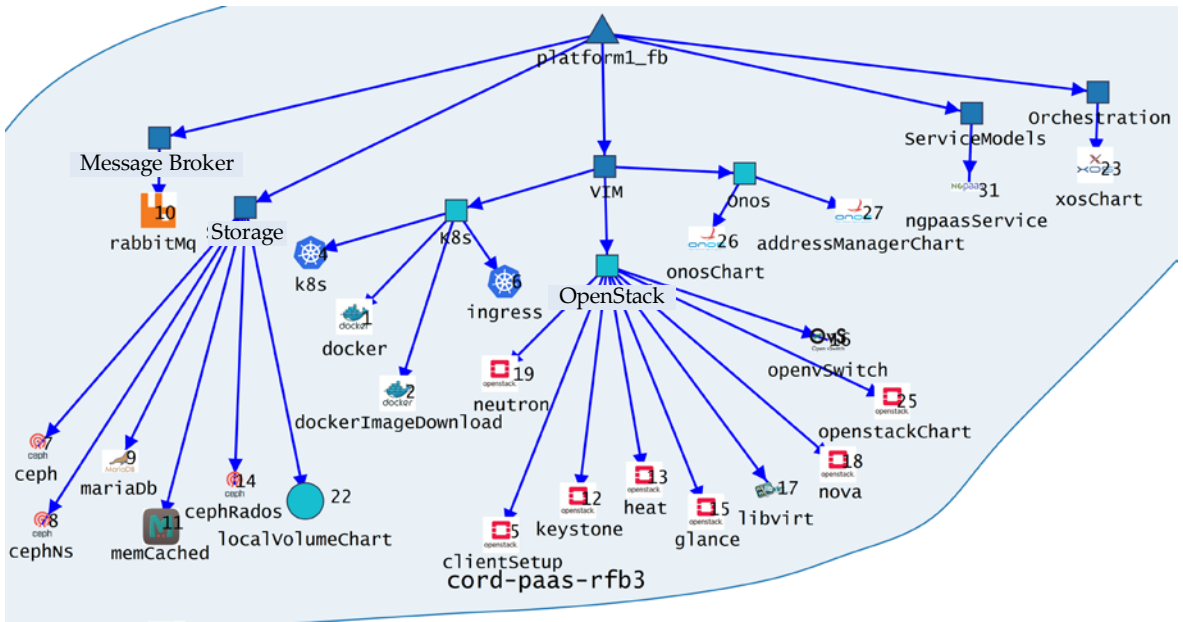409 graph that composes the CORD 6.0 platform in the RDCL 3D web interface.

410

411 Figure 13: RFBization of the CORD platform via RDCL 3D

412 As can be observed in **Figure 13**, the RFBs have been grouped into five major categories, as
413 children of the root RFB; depending on the role of the component they are responsible for deploying.
414 These five categories are 1) Message Brokers, 2) Storage Mechanisms, 3) Virtual Infrastructure
415 Managers (VIMs), 4) Orchestration and 5) Service Models. Additionally, the VIM group of RFBs is
416 also comprised of sub-groups, namely 1) Kubernetes 2) OpenStack and 3) ONOS. While this grouping
417 does not serve any functional role, it can significantly simplify the composition of RFB graphs. All
418 the remaining RFBs are leaf RFBs, meaning that they are directly mapped to the
419 execution/deployment of a micro-service. It can also be observed that all leaf RFBs are associated with
420 a priority value, ensuring that their deployment in the execution environment is done in a preferred
421 order. **Table 4** provides a detailed list of all leaf RFBs related to the deployment of the CORD
422 platform.

423

424

425

426

427

428   Table 4: List of Deployment RFBs

RFB Name	RFB Role
k8s	Deploys *Kubernetes*, which manages all Docker containers comprising CORD.
ingress	Deploys *Ingress*, a component related to Kubernetes which facilitates external access to services running in the cluster.
rabbitMq	Deploys *RabbitMQ*, the message broker used in OpenStack
ceph	A set of RFBs that deploy *ceph*, the distributed storage component of OpenStack
marriaDB	Deploys *marriaDB*, the database for OpenStack
memCached	Deploys *memcached*, the distributed memory caching system for OpenStack
heat	Deploys *Heat*, the OpenStack orchestrator.
glance	Deploys *Glance*, the OpenStack image registry.
keystone	Deploys *Keystone*, the OpenStack identity service.
openvSwitch	Deploys an *OpenvSwitch*, instance that connects all OpenStack VMs.
nova	Deploys *Nova*, the computing service of OpenStack.
neutron	Deploys *Neutron*, the networking service of OpenStack. In CORD Neutron is responsible for creating the networks connecting the different VMs.
openstackChart	Deploys the *OpenStack service synchronizer* of CORD.
addressManagerChart	Deploys a set of components, which enable public connectivity for VNFs.
onosChart	Deploys the *ONOS synchronizer* in CORD.
xosChart	Deploys the *XOS synchronizer* in CORD.
ngpaasserviceChart	Deploys the *ngpaasService synchronizer*. It also onboards the service model.
localVolumeChart	Deploys the persistent storage component of CORD.
clientSetup	Deploys the *OpenStack client* (Command Line Interface).
libvirt	Deploys *libvirt*, the virtualization service to be used by OpenStack
docker	Deploys *Docker* on the target node.
dockerImageDownload	Downloads all CORD related *Docker Images*

429   *5.2. RFBization of Services and Value-Added Services*

430   Following a similar approach to the RFBization of the CORD platform, all services and VAS
431   supported by the Telco PaaS have also been partially or fully RFBized and integrated into the NGPaaS
432   workflow. With regards to the deployment of virtual Firewalls, Routers and monitoring probes two
433   distinct RFB types are available 1) Instance RFB, which deploys VM based services in the OpenStack
434   environment of CORD and 2) Network RFB, which deploys networks in the Neutron environment of

435 CORD. Using these two RFB types, complex deployment scenarios can be expressed by the VSP.
436 **Figure 14** illustrates an RFB graph comprising of three RFB groups 1) Router 2) Firewall and 3) Probe
437 and 5 RFB leaves 1) Router Instance 2) Firewall Instance 3) Probe Instance 4) Data Network and 5)
438 Monitoring Network. Following the associations between the different RFB groups and leaves, this
439 RFB graph will provide a service chain of a Firewall and Router VNF (through a commonly shared
440 data network), in addition this graph also expresses that out of the two VNFs the Router is to be
441 monitored by a probe, via the monitoring network. The side-car probes of the monitoring/healing
442 infrastructure have been RFBized, while not available at this point, it is also possible to RFBize the
443 remaining components (e.g. ELK stack, Healer).

444 Additionally, the service provider can deploy service instances, networks and connectivity
445 requests agnostically to the underlying technologies. It is the role of the NGPaaS OSS/BSS to translate
446 these requests to the technology-specific APIs. More specifically for the case of the deployment of
447 Figure 14, each of the leaf RFBs will be translated to a TOSCA recipe and then published to the
448 TOSCA endpoint of the XOS orchestrator. Then XOS will decide how to serve this request best and
449 translate this set of TOSCA recipes into API calls to OpenStack and ONOS. The API calls to
450 OpenStack Nova will create the VM instances that will hold the router and firewall VNFs, while API
451 calls to OpenStack Neutron will create the monitoring and data networks. Finally, through the
452 dedicated REST API, XOS will instruct ONOS to populate the network with the necessary OpenFlow
453 rules to facilitate connectivity between the VNFs.

454 In the Telco PaaS, the VM images used to instantiate the VNFs have all been preconfigured with
455 an SSH channel, which provides access to their CLI interface. This way any entity with access to the
456 deployed VNFs (e.g. VSP, end user) could interface with them and configure them on demand,
457 without the need to re-provision them. However, if more virtual resources are required (e.g. more
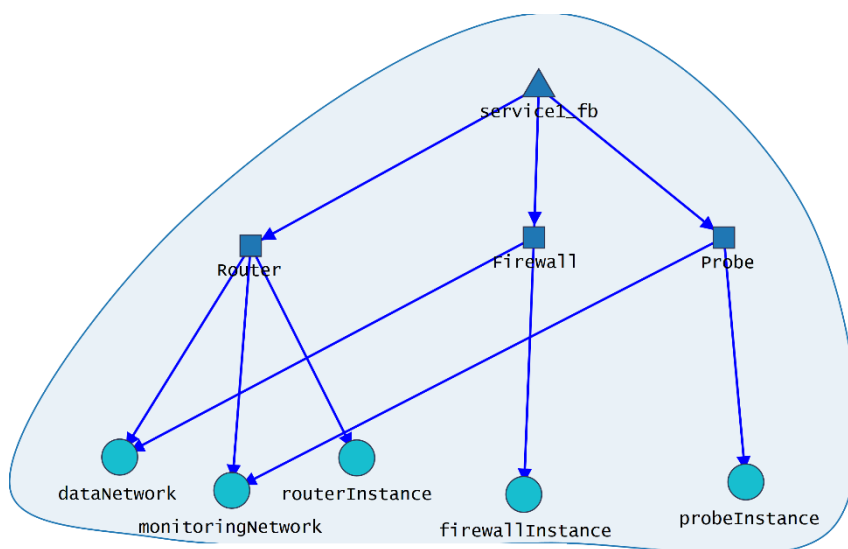458 vCPUs), then a new VNF deployment will be necessary using the NGPaaS workflow.

459

460 Figure 14: RFBization of service deployment

461 The network policy framework has also been RFBized, thus allowing a VSP to either deploy
462 policy type implementations in the policy framework or request for the enforcement of policy
463 instances in the network. With regards to deploying the components of the policy framework only
464 one RFB needed to be defined, namely the ONOSapp RFB. Deploying RFB graphs comprising of
465 ONOSapp RFBs will result in the desired ONOS applications to be installed and activated on the
466 ONOS SDNC of the CORD platform. In **Figure 15** (left), an RFB graph that requests the deployment
467 of three ONOS applications is illustrated. These applications include the Policy Manager of the policy

468 framework and two policy type implementations (Firewall and NAT). Once this RFB graph has been
469 deployed, then the VSP can request the enforcement of Firewall and NAT policy instances. Following
470 the example of **Figure 12**, the right side of **Figure 15** illustrates an RFB graph that requests the
471 enforcement of a Firewall policy instance. To facilitate the creation of RFB graphs of this type a
472 collection of RFB types is available, one for each supported policy type (e.g. Firewall Policy Instance).

473 Same as with the deployment of the service graph of **Figure 14**, the deployment of network
474 policies through the RDCL 3D tool is also technology agnostic. The service provider need only
475 provide the necessary metadata with the deployment request (e.g. endpoints to block with the
476 firewall policy). It is the role of the RDCL 3D tool and agent to translate this request to a JSON string
477 and ship to the ONOS SDNC. After that, the policy framework within ONOS will validate the policy
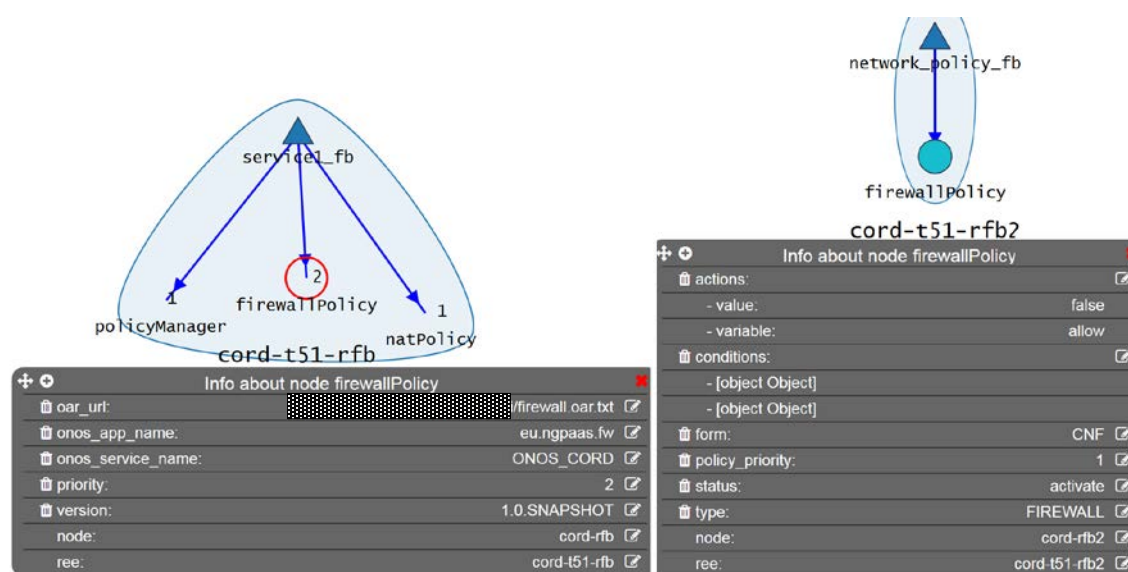478 request and install the required OpenFlow rules in the network.

479

480 Figure 15: RFBization of the policy framework

481 **6. 5G Public Safety use-case**

482 As discussed, because the NGPaaS workflow is based on the concepts of build-ship-run and
483 build-to-order, it allows for the deployment of a wide-range of platforms, services and technologies.
484 So far, this paper focused on the Telco PaaS use case and the VNFaaS PoC. However, to demonstrate
485 the flexibility of the NGPaaS workflow another use case has also been considered, namely the 5G
486 Public Safety use case. This use case is briefly demonstrated herein, through the Mission Critical Push
487 To Talk (MCPTT) PoC.

488 The idea of the 5G Public Safety use case is that an end-to-end cloud native mobile network
489 supporting the MCPTT service is provisioned on-demand, through NGPaaS. Then it can be used by
490 firemen during an intervention at an emergency incident (e.g. a building fire). Depending on the
491 location of the Radio Access Network (RAN), this use case could involve different scenarios. In the
492 prototype presented herein, the fire truck is assumed to have its own, mini, data centre with the
493 capability to host VNFs. In this way the U-Plane as well as the Cloud RAN and MCPTT related VNFs
494 can be hosted in the fire truck, while the MCPTT control plane VNFs can be hosted in a centralised
495 data centre. In this PoC instead of CORD, two separate Kubernetes platforms are deployed: A Core
496 Kubernetes PaaS and an Edge Kubernetes PaaS, tailored respectively for the Core and RAN service
497 requirements. Both Kubernetes platforms are fully deployable and configurable through the NGPaaS
498 workflow, the same workflow used for the Telco PaaS use case. Finally, the VNFs comprising the
499 MCPTT PoC are deployed on their corresponding platforms, similarly to that of the VNFaaS PoC.

500 **Figure 16** illustrates the architecture and functional elements of this PoC. More details are available
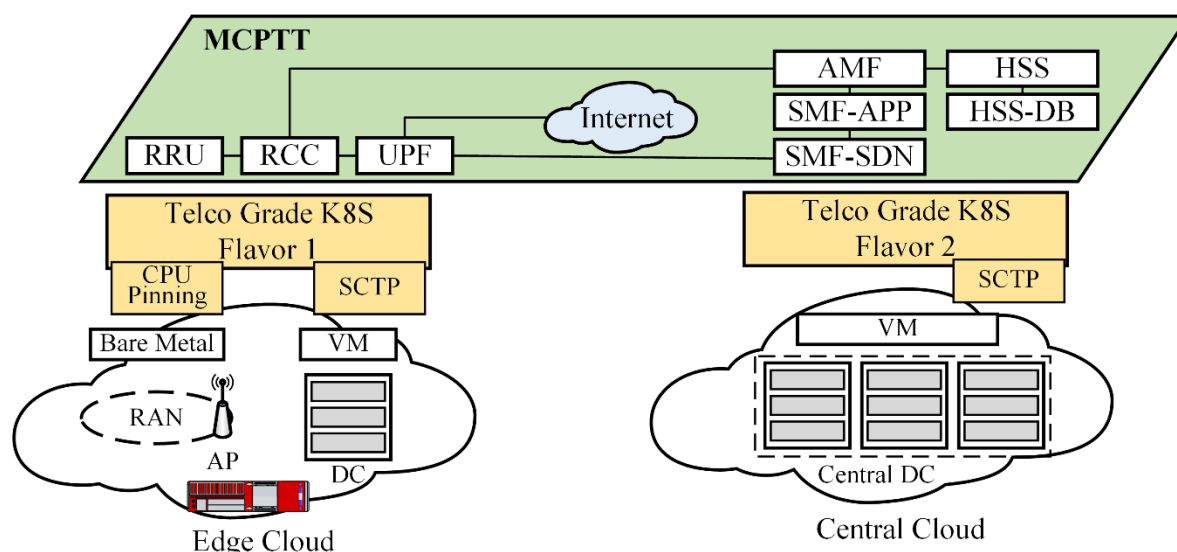501 in the technical documentation of the NGPaaS [37].

502

503 Figure 16: The 5G Public Safety Use Case

504 **7. Discussion and Conclusions**

505 This paper highlights some of the inherent complexities and challenges associated with service
506 and platform deployment in cloud-based environments, as required for the successful adoption of
507 5G technologies. This complexity can be mainly attributed to the broad spectrum of available
508 infrastructure and platform technologies, each of which brings their own interfaces and
509 communication protocols. As a possible solution to this problem, this paper introduced a novel
510 workflow for the composition and deployment of platforms and services in multi-cloud
511 environments. The proposed workflow follows the NGPaaS concepts of build-to-order and Build-
512 Ship-Run, by utilising the RFB model and the RDCL 3D tool. Build-to-order implies the possibility
513 to define custom platforms and services agnostically to the underlying technologies, based on
514 customer demand. Build-Ship-Run implies the possibility to deploy this platform or service
515 compositions on a wide range of execution environments, through a centralised environment (RDCL
516 3D tool). By following the NGPaaS workflow, platform and service deployment is simplified greatly
517 and facilitated, since most of the underlying technologies are abstracted through generic blueprints
518 and APIs. Finally, this paper successfully validated the proposed workflow by presenting a proof of
519 concept scenario (Telco PaaS) that includes the composition and deployment of both a platform and
520 a set of diverse services. The Telco PaaS use case is based on the CORD platform and represents a
521 VNFaaS scenario, in which a VSP offers VNF services to its end customers, together with many value-
522 added services (monitoring, healing, policy-based network management). Finally, to showcase the
523 flexibility of the NGPaaS workflow, a second use case is also, presented herein.

524 **Author Contributions:** Conceptualization, Paul Veitch and Bessem Sayadi; Investigation, Angelos Mimidis-
525 Kentis, Jose Soler, Paul Veitch, Adam Broadbent, Marco Mobilio, Steven Van Rossem and Bessem Sayadi;
526 Methodology, Angelos Mimidis-Kentis, Jose Soler, Paul Veitch and Adam Broadbent; Project administration,
527 Jose Soler and Bessem Sayadi; Resources, Jose Soler, Paul Veitch, Oliviero Riganelli, Wouter Tavernier and
528 Bessem Sayadi; Software, Angelos Mimidis-Kentis, Adam Broadbent, Marco Mobilio and Steven Van Rossem;
529 Supervision, Jose Soler, Paul Veitch, Oliviero Riganelli, Wouter Tavernier and Bessem Sayadi; Validation,
530 Angelos Mimidis-Kentis, Paul Veitch, Adam Broadbent, Marco Mobilio and Steven Van Rossem; Visualization,
531 Angelos Mimidis-Kentis; Writing – original draft, Angelos Mimidis-Kentis, Paul Veitch and Steven Van Rossem;

References

538 1. J. Erfanian et al., "Network Functions Virtualisation – White Paper on NFV priorities for 5G," ETSI White
539   Pap., no. 1, pp. 1–15, 2017.

540 2. F. Van Lingen et al., "The Unavoidable Convergence of NFV, 5G, and Fog: A Model-Driven Approach to
541   Bridge Cloud and Edge,"." IEEE Communications Magazine, vol. 55, no. 8, pp. 28–35, 2017.

542 3. Van Rossem, Steven, et al. "A Vision for the Next Generation Platform-as-a-Service." 2018 IEEE 5G World
543   Forum (5GWF). IEEE, 2018.

544 4. A. Mimidis, et al., "The Next Generation Platform as a Service Cloudifying Service Deployments in Telco-
545   Operators Infrastructure," in Proceedings of the 25th International Conference on Telecommunications
546   (ICT 2018), 2018.

547 5. Stefano Salsano et al., 'RDCL 3D, a Model Agnostic Web Framework for the Design and Composition of
548   NFV Services', 3rd IEEE International Workshop on Orchestration for Software Defined Infrastructures,
549   O4SDI at IEEE NFV-SDN conference, Berlin, 6-8 November 2017

550 6. L. Peterson, "CORD: Central Office Re-Architected as a Datacenter," Open Networking Lab white paper,
551   November, 2015.

552 7. J. Quittek, et al., "Network Functions Virtualisation (NFV); Management and Orchestration," Gs Nfv-Man
553   001 V1.1.1, vol. 1, pp. 1–184, 2014.

554 8. Röck, C., and S. Kolb. "Nucleus–Unified Deployment and Management for Platform as a Service."
555   University of Bamberg, Tech. Rep (2016): 49.

556 9. Van Rossem, Steven, et al. "Introducing Development Features for Virtualized Network Services." IEEE
557   Communications Magazine 99 (2018): 2-10.

558 10. ETSI NFV ISG, "Network Functions Virtualisation (NFV); Network Service Templates Specification", ETSI
559   GS NFV-IFA 014 V2.1.1 (2016-10)

560 11. Giuseppe Bianchi et al., "Superfluidity: a flexible functional architecture for 5G networks," Trans. Emerg.
561   Telecommun. Technol., vol. 27, no. July, pp. 1178–1186, 2016.

562 12. Veitch, Paul, et al. "Re-Factored Operational Support Systems for the Next Generation Platform-as-a-
563   Service (NGPaaS)." 2018 IEEE 5G World Forum (5GWF). IEEE, 2018.

564 13. Berde, Bela, et al. "Dev-for-Operations and Multi-sided Platform for Next Generation Platform as a Service."
565   2018 European Conference on Networks and Communications (EuCNC). IEEE, 2018.

566 14. "Elastic stack." [Online]. Accessed 12-12-2018: https://www.elastic.co/.

567 15. "Cisco   Cloud   Service   Router   100V   series",   [Online].   Accessed   01-04-2019,
568   https://www.cisco.com/c/en/us/products/routers/cloud-services-router-1000v-series/index.html

569 16. "Fortinet-   FortiGate   Virtual   Appliance"   [Online].   Accessed   01-04-2019,
570   https://www.fortinet.com/content/dam/fortinet/assets/data-sheets/FortiGate_VM.pdf

571 17. "The Helm package manager for Kubernetes" [Online]. Accessed 01-04-2019, https://helm.sh/

572 18. Ferran Canellas, et al. "Policy Framework Prototype for ONOS", 2019 IEEE Conference on Network
573   Softwarization (NetSoft). IEEE, 2019 (Accepted).

574 19. "Open Source MANO" [Online]. Accessed 01-04-2019, https://osm.etsi.org/

575 20. "5G Tango" [Online]. Accessed 01-04-2019, https://www.5gtango.eu/

576 21. "Kubernetes" [Online]. Accessed 01-04-2019, https://kubernetes.io/.

577 22. " Red Hat Ansible" [Online]. Accessed 01-04-2019, https://www.ansible.com/

578 23. Nogales, Borja, et al. "Design and Deployment of an Open Management and Orchestration Platform for
579   Multi-Site NFV Experimentation." IEEE Communications Magazine 57.1 (2019): 20-27.

580 24. Yilma, Girma M., et al. "On the Challenges and KPIs for Benchmarking Open-Source NFV MANO Systems:
581   OSM vs ONAP." arXiv preprint arXiv:1904.10697 (2019).

582 25. Alvarez, Federico, et al. "An Edge-to-Cloud Virtualized Multimedia Service Platform for 5G
583   Networks." IEEE Transactions on Broadcasting (2019).

584 26. Tarafdar, Naif, et al. "Building the Infrastructure for Deploying FPGAs in the Cloud." Hardware
585   Accelerators in Data Centers. Springer, Cham, 2019. 9-33.

586 27. S. Chiotakis, S. et al. "vFPGAmanager: A Hardware-Software Framework for Optimal FPGA Resources
587   Exploitation in Network Function Virtualization," 2019 European Conference on Networks and
588   Communications (EuCNC), 2019.

589 28. Li, Xiaoyao, et al. "DHL: Enabling Flexible Software Network Functions with FPGA Acceleration." 2018
590   IEEE 38th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2018.

591 29. Zhang, Qixia et al. "Adaptive Interference-Aware VNF Placement for Service-Customized 5G Network
592   Slices." IEEE INFOCOM 2019-IEEE Conference on Computer Communications. IEEE, 2019.

593 30. Fei, Xincai, et al. "Adaptive vnf scaling and flow routing with proactive demand prediction." IEEE
594   INFOCOM 2018-IEEE Conference on Computer Communications. IEEE, 2018.

595 31. Wang, Tao et al. "Multi-resource load balancing for virtual network functions." 2017 IEEE 37th International
596   Conference on Distributed Computing Systems (ICDCS). IEEE, 2017.

597 32. Fei, Xincai, et al. "Towards load-balanced vnf assignment in geo-distributed nfv infrastructure." 2017
598   IEEE/ACM 25th International Symposium on Quality of Service (IWQoS). IEEE, 2017.

599 33. Miao, Wang, et al. "Stochastic Performance Analysis of Network Function Virtualization in Future
600   Internet." IEEE Journal on Selected Areas in Communications 37.3 (2019): 613-626.

601 34. Mfula, Harrison, and Jukka K. Nurminen. "Self-Healing Cloud Services in Private Multi-Clouds." 2018
602   International Conference on High Performance Computing & Simulation (HPCS). IEEE, 2018.

603 35. Li, Zhijing, et al. "Predictive Analysis in Network Function Virtualization." Proceedings of the Internet
604   Measurement Conference 2018. ACM, 2018.

605 36. Acker, Alexander, et al. "Online Density Grid Pattern Analysis to Classify Anomalies in Cloud and NFV
606   Systems." 2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom).
607   IEEE, 2018.

608 37. "NGPaaS Deliverable 5.3: Initial platform prototype for 5G PaaS", June 2018

609